VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2012

Tomáš Machala

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Machala**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: NetDirect s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec**

Konzultant bakalářské práce: Bc. Radka Radecká

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012

.....

Rád bych na tomto místě poděkoval společnosti NetDirect s.r.o. za možnost absolvování odborné praxe, Bc. Radce Radecké, v jejímž týmu jsem pracoval a Ing. Petrovi Chovancovi za vedení mé práce.

Abstrakt

Cílem této bakalářské práce je popsat průběh mé praxe ve společnosti NetDirect s.r.o., dodavatele e-business aplikací s významným tržním podílem na českém a slovenském trhu. Během praxe jsem se podílel na vývoji nové generace e-commerce platformy.

Práce je rozdělena do tří částí. V první části je představena společnost, její hlavní produkty a mé pracovní zařazení. Druhá část popisuje softwarovou architekturu projekt, agilní metodiku Scrum a úkoly, které jsem během praxe řešil. Poslední část obsahuje zhodnocení dosažených výsledků a zkušeností, které mi praxe přinesla.

Klíčová slova: Odborná praxe, NetDirect, e-commerce, .NET, vývoj aplikací, agilní vývoj, Scrum

Abstract

The thesis aims on description of my practice in NetDirect s.r.o., a supplier of e-business applications with a significant market share in the Czech and Slovak markets. I was assigned to a research team responsible for development of a new generation of e-commerce platform.

The thesis is divided into three parts, first of which describes the company, its main products and my assignment. The second part describes the project architecture, Scrum (an agile software development method we were using) and the tasks I was resolving. The last part concludes achieved results and experiences I have gained.

Keywords: Professional practice, NetDirect, e-commerce, .NET, application development, agile development, Scrum

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
CMS	–	Content Management System
CRUD	–	Create, Read, Update, Delete
CSS	–	Cascading Style Sheets
CVS	–	Concurrent Versions System
HTML	–	HyperText Markup Language
HTTP	–	Hypertext Transfer Protocol
IoC/DI	–	Inversion of Control/Dependency Injection
JSON	–	JavaScript Object Notation
LINQ	–	Language Integrated Query
ORM	–	Object-Relational Mapping
SEO	–	Search Engine Optimization
SQL	–	Structured Query Language
T-SQL	–	Transact-SQL
TFS	–	Team Foundation Server
WCF	–	Windows Communication Foundation
XML	–	Extensible Markup Language
XSL	–	Extensible Stylesheet Language
XSLT	–	Extensible Stylesheet Language Transformations

Obsah

Seznam obrázků

1 Úvod

Rozhodl jsem se absolvovat bakalářskou práci formou odborné praxe, protože věřím, že je pro mě taková forma přínosnější než samostatné bádání nad zadaným tématem. Zajímám se o návrh informačních systémů, webové aplikace, databáze a systémovou integraci. Rád bych se těmto oborům věnoval i v budoucnu a proto jsem hledal firmu, ve které by mě čekala práce blízká těmto tématům. Našel jsem společnost NetDirect s.r.o., kde jsem byl na základě přijímacího pohovoru přijat a zařazen do vývojového týmu.

V této práci, rozdělené do tří částí, budu popisovat průběh mé praxe v NetDirectu. V první představím společnost, její hlavní produkty a mé pracovní zařazení. Druhá část popisuje akchitekturu projektu, agilní metodiku Scrum pro řízení projektů a úkoly, které jsem během praxe řešil. Poslední část obsahuje subjektivní zhodnocení dosažených výsledků a zkušeností, které mi praxe přinesla.

2 Představení společnosti

NetDirect s.r.o. je dodavatel e-business aplikací s asi 100 zaměstnanci a významným tržním podílem na českém a slovenském trhu. Společnost je držitelem titulu *Microsoft Gold Certified Partner* a umístila se na předních místech v řadě soutěží – získala tři první místa na světové konferenci *WPC09* v USA a nejvyšší ocenění na partnerské konferenci *WPC10*, kde byla vyhlášena nejlepším partnerem roku a získala titul *2010 Microsoft Country Partner of the Year*.

2.1 Hlavní produkty

Stěžejní produkty, které NetDirect svým zákazníkům nabízí, jsou internetové obchody FastCentrik a ShopCentrik a CMS MediaCentrik.

2.1.1 FastCentrik

FastCentrik je „krabicové“ řešení internetového obchodu, poskytované jako služba, za kterou zákazník platí formou paušálních poplatků. Službu lze objednat ve třech variantách, které se liší funkcionalitou a cenou.

2.1.2 ShopCentrik

ShopCentrik je naopak řešení dodávané zákazníkům na klíč od analýzy, grafiky až po nasazení do ostrého provozu. Každé řešení je přizpůsobeno s ohledem na cílové skupiny a prodávané komodity.

Významnou vlastností obou systémů je integrace Aukro Konektoru – systému, který vznikl v rámci spolupráce NetDirectu s Aukro.cz a který umožňuje automaticky vystavovat zboží z e-shopu i na Aukru.

2.1.3 MediaCentrik

Třetím produktem je CMS MediaCentrik, na kterém lze provozovat webové stránky, případně tzv. microsites. Pro představu bych jej přirovnal k open-source řešením jako Joomla nebo Wordpress. Je navržen s důrazem na SEO a snadnou administraci.

2.2 Mé pracovní zařazení

Po přijímacím pohovoru jsem byl zařazen jako programátor do tehdy sedmičlenného vývojového týmu. Mému umístění nejspíš pomohlo, že jsem již s vývojem informačních systémů měl praktické zkušenosti. Tým se po dobu mé praxe postupně rozrůstal o nové lidi a na konci praxe už nás bylo přes 20. Naší zodpovědností byl vývoj nové e-commerce platformy, která bude v budoucnu sloužit jako framework, nad kterým se budou implementovat nové projekty pro zákazníky.

3 Představení projektu

3.1 Architektura projektu

Systém je z velké části postavený a hostovaný na technologiích Microsoftu. Běží na systémech Windows Server, jako databázi používá SQL Server 2008 a aplikační logika je napsaná v jazyce C#. Systém můžeme rozdělit na tři vrstvy – prezentační, aplikační a databázovou.

3.1.1 Prezentační vrstva

Prezentační vrstvu tvoří webové aplikace, se kterými pracují uživatelé systému – rozhraní samotného internetového obchodu pro zákazníky a administrační rozhraní pro provozovatele obchodu.

Na straně serveru jsme používali ASP.NET MVC 3. Místo výchozího šablonovacího systému Razor jsme používali vlastní řešení, založené na XSL šablonách. Na straně prohlížeče pak technologie související s HTML5 a JavaScriptové frameworky jQuery a Knockout. O prezentační vrstvě budu dále mluvit jak o front-endu.

3.1.2 Aplikační vrstva

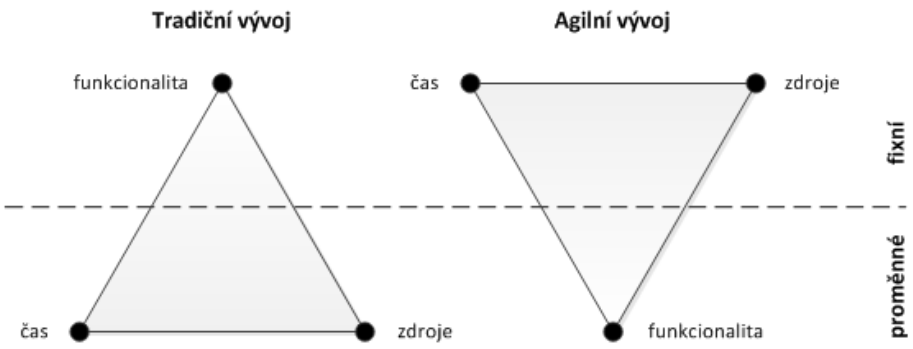
V aplikační vrstvě byla tvořena v jazyce C#. Obsahovala mimo jiné moduly, zodpovědné za určitou část systému – např. zboží, slevové kupónu, záruky, ankety, články... Na jedné straně aplikační vrstvy byla ORM vrstva, která zajišťovala výměnu dat s databází a na druhé straně WCF služba, která zpřístupňovala metody z jednotlivých modulů vyšším vrstvám, především administraci. Aplikační vrstvu budu dále nazývat back-endem.

3.1.3 Datová vrstva

Data jsou standardně uložena v databázi SQL Server 2008. Jednotlivé moduly aplikační vrstvy je ale možné přepsat tak, aby pracovaly i s jinými zdroji dat, pokud by to nějaký budoucí projekt postavený na našem projektu vyžadoval. K datům se přistupuje striktně jen a pouze přes uložené procedury.

3.2 Metodika Scrum

NetDirect byl jednou z prvních českých IT společností, která pro řízení projektů zavedla používání agilní metodiky Scrum. Tuto metodiku používal i náš tým. Jeho základní myšlenkou je, stejně jako u ostatních agilních metodik, připravenost na změny. Předpokládá, že se během životního cyklu projektu budou funkční požadavky výrazně měnit; např. kvůli změnám na trhu nebo novým poznatkům, které z projektu vyplynou. Za fixní naopak považuje přidělené zdroje a čas. Tradiční metody vývoje softwaru se k těmto třem prostředkům staví přesně naopak – drží se předem specifikovaných funkčních požadavků a přizpůsobují zdroje a čas.



Obrázek 1: Tradiční vs. agilní vývoj softwaru

Očekávaným změnám je podřízen způsob práce, který se neopírá o žádnou důkladnou funkční analýzu. Vývoj softwaru je plánován iterativně a po malých přírůstcích, které se průběžně prezentují zákazníkovi. Na základě jeho zpětné vazby se pak přizpůsobují plány dalšího vývoje. Jelikož byl tento projekt interní, vystupoval v roli zákazníka náš management.

Pro mě, jako pro člena týmu, znamenalo používání metodiky Scrum především dvě věci:

1. Účastnění se tzv. „stand-upů“, tedy krátkých pravidelných setkání, na kterých se probírá aktuální stav úkolů; jestli něco brání jejich dokončení a pokud ano, tak co se dá udělat pro odstranění překážky.

2. Intenzivní používání unit testů, které jsou u agilních metod kvůli častým změnám v kódu velmi zásadní.

Poznámka 3.1 Stand-upy by správně měly probíhat každý den a trvat maximálně 10 min. Účastníci by při nich skutečně měli stát, aby neměli tendenci zbytečně je protahovat. Takto ortodoxně jsme ale pravidla nedodržovali.

3.3 Práce se sdíleným zdrojovým kódem

Jelikož náš tým potřeboval pracovat nad společným zdrojovým kódem, používali jsme Microsoft Team Foundation Server. Princip práce s TFS je podobný, jako s jinými CVS (např. Apache Subversion). Má ale výhodu v nativní podpoře ze strany Visual Studia.

Na začátku práce jsem si vždy stáhnul aktuální verzi kódu, upravil jej a nakonec provedl tzv. „check-in“ – tedy nahrál změny zpět na server. TFS umí do určité míry samo řešit konflikty, které vzniknou, když více uživatelů nezávisle na sobě upraví ten samý soubor a nahrají jej zpátky. V takovém případě soubory vzájemně porovná a pokusí se změny sloučit.

U některých částí projektu jsme měli nastavený tzv. „kontrolní build“. Chtěl-li někdo provést check-in, TFS se nejdříve pokusilo nahrávaný projekt sestavit. Pokud sestavení skončilo chybou, byly změny odmítnuty. Odmítnuté změny bylo na serveru možné pouze

uložit pro případ ztráty dat, ale neovlivnily samotný projekt. Smyslem kontrolního buildu bylo, aby na server nikdo nenahrával nesestavitelné verze, které by ostatním bránily v práci.

3.4 Plánování úkolů

Náš tým používal TFS také jako nástroj k podpoře metodiky Scrum. Evidovali jsme v něm úkoly a vykazovali čas, který jsme na nich strávili. Vždy, když jsme dělali check-in, jsme navíc mohli označit úkoly, kterých se týkal.

4 Řešené úkoly

Úkoly, které jsem řešil, bych časově rozdělil do tří etap:

1. V první jsem řešil různé nesouvisející drobnosti, díky kterým jsem se měl seznámit s architekturou projektu.
2. Většinu času stráveného na praxi jsem strávil druhou fází, která se týkala implementaci modulů na aplikační vrstvě. Vytvářel jsem části ORM, rozšiřoval WCF službu o nové metody a testoval svou práci pomocí unit testů. Napsal jsem si i pár T-SQL procedur, což sice nebylo mou náplní práce, ale občas jsem si takto pomáhal, když databázoví specialisté zrovna neměli čas na úpravy, které jsem potřeboval.
3. Poslední etapu mé praxe jsem strávil v týmu, který implementoval front-endovou část. Vytvářeli jsme internetový obchod s elektronikou „ElektroCentrik“. V budoucnu bude sloužit jako jedna z předpřipravených grafických šablon, kterou si zákazníci budou moct pro svůj obchod vybrat. Často jsem pracoval s moduly, které jsem vytvořil v předchozí fázi.

Během druhé a třetí fáze jsem si vyzkoušel dva různé přístupy k vývoji softwaru. Zatímco back-end byl postavený na impozantně čisté a promyšlené architektuře, využívající složité návrhové vzory, práce na font-endu byla celkem přímočará a více než na nějakou architekturu spoléhala na neustálý refactoring. Osobně mi byl bližší ten první přístup.

4.1 Moduly Články a Kategorie článků

Tímto úkolem jsem se začal seznamovat s architekturou projektu. Spočíval ve vytvoření funkcionality pro články a jejich kategorie. Začal jsem přečtením specifikace, což se později ukázalo jako obecně dobrý první krok. V databázi již byly připraveny všechny potřebné procedury pro CRUD operace, získání seznamu článků v kategorii atd. Má práce spočívala ve vytvoření metod na aplikační vrstvě, které tyto uložené procedury využívaly a v implementaci komponent pro zobrazení na front-endu.

4.1.1 Vytvoření modulu

Podle vzorů, které jsem postupně odkoukal z již existujících modulů, jsem vytvořil mimo jiné třídu, reprezentující danou entitu, rozhraní definující metody tohoto modulu a repository, což je implementace této služby a spolu s dalšími objekty tvoří ORM vrstvu, řešící výměnu dat s databází. Metody definované v rozhraní jsem přidal také do kontraktů pro WCF službu. Jinak by nebyly přístupné z administrace. Nakonec jsem všechny vytvořené objekty okomentoval a napsal pro modul unit testy.

Práci jsem nahrál zpět do repository na TFS a naplánoval build. Poté jsem ve Visual Studiu otevřel solution s front-endovou částí a stáhnul aktuální verzi kódu a sestavených DLL knihoven. Toto bylo obecně nutné dělat vždy, když jsem potřeboval, aby se změny provedené v jedné solution projevily i v ostatních. Projekt byl totiž kvůli velikosti rozdělen na mnoho menších částí, které se kompilovaly do samostatných DLL knihoven.

4.1.2 Implementace na front-endu

Na front-endu zbývalo vytvořit stránky zobrazující detail článku a jejich seznam v kategorii. Použil jsem přitom náš vlastní šablonovací systém, založený na XSLT.

4.2 Benchmark uložených procedur

Databázová vrstva obsahovala několik uložených procedur, které vracely informace o komoditách na základě jejich ID. Všechny procedury vracely stejná data, lišily se ale vnitřní implementací. Cílem mého úkolu bylo vytvořit testovací nástroj pro automatické porovnávání jejich výkonu. Naměřená data pak používali databázoví specialisti při rozhodování, které implementace zahodit a které vypadají nadějně a má smysl na nich dále pracovat.

Optimalizace této konkrétní funkce měla velký význam, protože zobrazení vyfiltrovaných komodit je u internetového obchodu jedna z nejčastěji volaných operací, jejíž výsledky se kvůli různorodosti vstupních parametrů navíc nedají moc dobře cachovat.

Vytvořil jsem tedy konzolovou aplikaci, která procedury opakovaně volala pro různý počet komodit a zaznamenávala časy. Chtěl jsem naměřené hodnoty vizualizovat přehledněji, než jen vypsáním do konzole a proto jsem aplikaci rozšířil o generování HTML reportu. Z naměřených hodnot se pro větší přesnost pomocí LINQ výrazu spočítal medián, v paměti se vytvořil XML dokument s daty a ten se pomocí XSL stylu transformoval do výstupního HTML.

Z testů vyplynulo, že o proceduře často nejde jednoznačně říct, jestli je dobrá nebo špatná. Některé procedury, které byly nejrychlejší při zpracování malého množství komodit, rychle ztrácely výkon s tím, jak komodit přibývalo a naopak.

4.3 Univerzální pager

Můj další úkol spočíval v návrhu a implementaci pageru, který měl být univerzálně použitelný ke stránkování různých komponent. Z pohledu uživatele šlo o známý proužek s odkazy na jednotlivé stránky.

Chtěl jsem, aby pomocí pageru bylo možné stránkovat i komponenty vytvořené v budoucnu, aniž by se musel upravovat. Proto jsem jej navrhnul tak, aby všechny komponenty, které jej chtějí využívat, musely implementovat rozhraní IPageableComponent, skrz které pager získával všechny potřebné informace. Pager měl možnost nastavení maximálního počtu boxů zobrazených vedle sebe. Bylo-li stránek více, než počet povolených boxů, vykresloval je v sekcích oddělených třemi tečkami.

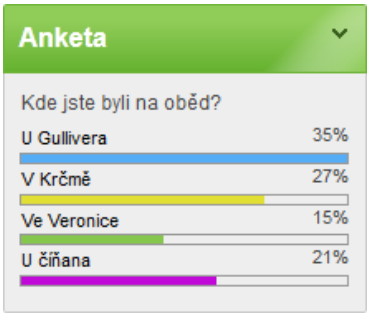
4.4 Modul Ankety

Modul ankety slouží k získávání zpětné vazby od zákazníků. Zákazníci jej vidí jako klasický box s otázkou a několika možnostmi, pro které můžou hlasovat. Ankety měly podle specifikace umožňovat zadání otázky a odpovědí ve více jazycích. Pro každou anketu měly jít nastavit parametry jako časové rozmezí platnosti, způsob zobrazení hlasů

(počet nebo procenta), počáteční smyšlené hlasy pro jednotlivé odpovědi a další. Modul jsem implementoval jen na aplikační vrstvě. Zobrazení na front-endu řešil kolega.

4.4.1 Implementace modulu

Aby se ankety daly spravovat v administraci, musel modul podporovat CRUD operace. Kvůli výkonu jsem implementoval také vracení tabulkového výpisu anket pro administraci, kdy se nenačítaly celé entity, ale jen zjednodušený pohled na ně. Kvůli front-endu jsem implementoval ještě metodu pro přidání hlasu. Anketa (jako entita) se skládá z pěti tříd, které reprezentují její obecné vlastnosti jako časové rozpětí, kdy se má zobrazovat, otázky, jejich jazykové verze, odpovědi, opět jazykové verze a poslední uchovává informace o hlasech – datum, identitu nebo aspoň IP adresu hlasujícího atd.



Obrázek 2: Ukázka zobrazení ankety na front-endu

K uložení načtené entity zpět do databáze jsem využil schopnosti SQL Serveru 2008, přijmout jako parametr uložené procedury i tabulku. V ORM tak stačilo vytvořit několik objektů DataTable, naplnit je daty a uložit celou entitu včetně všech otázek i odpovědí pomocí jediné procedury. Procedura pak data pomocí funkce MERGE postupně roztřídila do tabulek, kam patří. Tento postup jsme se snažili používat vždy, nejen v případě anket. Měl dvě zásadní výhody:

1. ORM vrstva nemusela znát strukturu dat v databázi a tím pádem ji ani nebylo potřeba upravovat při změně struktury.
2. Z pohledu ORM byla celá operace atomická bez nutnosti použít transakci. Transakce se ve finále samozřejmě použila, ale teprve uvnitř uložené procedury. Navazování transakcí na úrovni ORM sebou totiž nese riziko, že pokud ORM uvízne na nějaké chybě a neprovede *commit* ani *rollback*, může databáze dlouho držet zámky na některé objekty, které by tak byly nepřístupné ostatním transakcím.

Naimplementované metody jsem přidal také do rozhraní pro WCF službu.

4.4.2 Unit Testy

Funkčnost své implementace jsem nakonec doladil pomocí unit testů. K testování CRUD operací jsem, nejen v tomto případě, používal postup, který elegantně otestuje všechny čtyři části (tedy create, read, update a delete):

1. Na aplikační vrstvě vytvořím novou instanci entity, vyplním všechny její atributy a uložím ji do databáze. Tím entita získá jednoznačné ID.
2. Načtu z databáze druhou instanci entity s tím samým ID a zkontroluji, že jsou všechny její atributy shodné s atributy první instance. Tím jsem ověřil funkčnost operací create a read.

3. Pak u jedné z instancí změním všechny atributy, které změnit lze, opět ji uložím, načtu zpátky a zkontroluji, že se atributy opravdu změnily. Tím je ověřena i operace update.

4. Zbývající operaci delete ověřím smazáním entity a zkontrolováním, že už ji nelze znovu načíst.

Jelikož je zvykem neměnit testováním data v databázi, psal jsem testy vždy tak, aby běžely v rámci transakce, která na konci provede rollback.

4.5 Modul Záruky

Modul záruky představuje číselník různých typů záruk. Jeho využití je především v kombinaci se zbožím. Každá záruka má definovanou jednotkou, což je také systémová entita a její velikosti. Jednotky stejného typu jsou vzájemně převoditelné. Díky tomu se dá záruka, definovaná v měsících, porovnávat např. se zárukou definovanou v letech.

4.5.1 Implementace modulu

Způsob implementace byl z velké části podobný jako u modulu Ankety. Díky chybě ve WCF službě, kterou jsem musel najít, jsem se seznámil s nástrojem WCF TestClient, který je dodávaný spolu s Visual Studiem. Umožňuje se připojit k WCF službě a testovat na ní volání metod. Během praxe se mi několik krát hodil.

4.6 Modul Kupóny

Kupón představuje poukaz na slevu, kterou může zákazník získat jako odměnu za nákup, v rámci reklamní akce apod. Je identifikován jednoznačným kódem a kromě něj má další parametry jako časové období platnosti, maximální počet použití, minimální cenu objednávky, od které jej lze uplatnit a další. Rozlišují se dva typy kupónů:

- **S fixní slevou** – sníží cenu objednávky o nastavenou hodnotu. Hodnota musí být nadefinována zvlášť pro všechny měny, ve kterých lze kupón použít.
- **S procentuální slevou** – stále stejná, nezávislá na měně, ve které zákazník nakupuje.

4.6.1 Implementace modulu

Na aplikační vrstvě jsem naimplementoval klasicky CRUD operace pro administrační rozhraní, zpřístupnil je i ve WCF službě a otestoval pomocí unit testů.

4.6.2 Implementace na front-endu

Na front-endu jsem rozšířil zobrazení nákupního košíku o box s políčkem pro zadání kódu. Po zadání se na straně serveru ověří, jestli je kód platný a jsou-li splněny i ostatní podmínky jeho použití, např. minimální cena objednávky.

Pokud použití kupónu nic nebrání, vloží se jako další položka do košíku. Naše implementace košíku totiž může obsahovat i jiné typy entit, než zboží. Na vložený kupón poté reaguje komponenta, vykreslující obsah košíku, která na základě vlastností kupónu zobrazí přepočítané ceny. Pokud kupón vložit nejde, zobrazí se zákazníkovi informace o chybě. Pro její zobrazení jsem použil modální dialog z jQuery.

4.7 Modul Hlídací pes

Hlídací pes je funkce, sloužící k hlídání ceny nebo skladové dostupnosti. Zákazník si jej může u konkrétního zboží aktivovat a nechat se upozornit e-mailem, jakmile dojde k jedné ze tří situací, kterou pes hlídá – pokles ceny pod nastavenou úroveň, jakýkoli pokles ceny nebo přijetí dříve nedostupného zboží na sklad. Aktivní hlídací psi představují také zdroj informací o zákaznících a proto je jejich seznam viditelný i v administraci. Hlídacího psa je možné navázat buď na e-mail, nebo na uživatelský účet. Díky tomu jej můžou používat i neregistrovaní uživatelé.

4.7.1 Implementace modulu

Implementace na aplikační vrstvě spočívala klasicky ve vytvoření CRUD operací, přidání metod do WCF služby a otestování pomocí unit testů. Funkčnost samotného sledování a odesílání e-mailů jsem neimplementoval, jelikož byla naplánována na později a během praxe jsem už ji nestihl. Měl jsem ale představu, že by aplikační vrstva periodicky (třeba co 4h) spouštěla v databázi proceduru, která by vrátila hlídací psy, u kterých došlo k události, kterou hlídají. Aplikační vrstva by uživatelům odeslala e-maily a entity by se poté odstranily z databáze.

4.8 Podpora pro osoby a jejich typy

Osoba je entita reprezentující fyzického člověka nebo firmu. Systém je používá na více místech, ale jejich hlavní smysl spočívá v tom, že můžou tvořit relaci se zbožím. Rozlišují se různé typy relací („autor“, „výrobce“. . .), které jsou definovány v číselníku typů osob.

Příklady:

- Osoba *„Lenovo“* je *„výrobce“* zboží *„ThinkPad T420“*
- Osoba *„Karel Čapek“* je *„ilustrátor“* zboží *„Povídání o pejskovi a kočičce“*

Zboží je pak podle osob, se kterými jsou v relaci, možné filtrovat. Na webu může být také abecední seznam osob s detaily pro každou z nich. Detail může zahrnovat popis, obrázek, odkaz na související zboží a další parametry. Stejně jako téměř všechny ostatní moduly jsou i osoby vícejazyčné.

4.8.1 Implementace modulu

Na aplikační vrstvě jsem opět klasicky vytvořil entity reprezentující osobu a číselník typů osob. Pro obojí jsem naimplementoval CRUD operace a několik dalších metod, vracejících

např. počet zboží, které je v relaci s danou osobou. Metody jsem zpřístupnil přes WCF službu a otestoval je pomocí unit testů.

4.8.2 Implementace na front-endu

Implementace front-endové části zahrnovala výpis osob, zobrazení detailu pro konkrétní osobu a seznam zboží dané osoby. Osoby ve výpisu se mi podařilo čistě pomocí XSLT seskupit do boxů podle prvního písmene a zobrazit v několika sloupcích. K větší univerzálnosti šablony jsem počet sloupců udělal parametrizovatelný.

4.9 Sledování odkazujících stránek

Další úkol spočíval v implementaci sledování stránek, ze kterých zákazníci do e-shopu přicházejí. Pokud pak během své session v systému vytvořili objednávku, adresa stránky, ze které původně přišli, se k ní uložila.

Adresu odkazující stránky lze identifikovat z hlavičky HTTP požadavku. Webové prohlížeče ji do nich obvykle přidávají jako parametr *„Referer“*:

```
GET http://www.mall.cz/apple-ipad/apple-ipad-2-16gb-wifi HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http :// tablety.heureka.cz/apple-ipad-2-16gb-wifi/
Accept-Language: cs-CZ
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Accept-Encoding: gzip, deflate
Host: www.mall.cz
Connection: Keep-Alive
```

Výpis 1: Ukázka HTTP hlavičky s parametrem *„Referer“*

Z této ukázkové hlavičky můžeme vyčíst, že se uživatel na stránku `mall.cz/apple-ipad/apple-ipad-2-16gb-wifi` dostal z porovnávače cen Heureka.cz, kliknutím na odkaz ze stránky `tablety.heureka.cz/apple-ipad-2-16gb-wifi/`.

Řešení tedy bylo jednoduché – pří vytvoření session jsem do ní uložil adresu odkazující stránky. Ta se pak v session držela celou dobu její životnosti. Pokud uživatel během této doby vytvořil objednávku, adresa se uložila do databáze, jako jeden z parametrů objednávky. Sbírání adres touto cestou ale určitě nebude příliš spolehlivé:

- Některé prohlížeče (např. Chrome a Safari) parametr Referer v hlavičce z důvodu ochrany soukromí defaultně neposílají.
- Parametr může být z hlavičky odstraněn i dodatečně – typicky firemním firewallem.
- Pokud uživatel neklikal na žádný odkaz, ale zadal adresu obchodu přímo z klávesnice, pak v hlavičce parametr „Referer“ také není, protože žádná odkazující stránka neexistuje. Tento stav navíc nelze odlišit od stavu, kdy odkazující stránka existuje, jen nebyla v hlavičce předána.

4.10 Našeptávač

Tento úkol spočíval v rozšíření vyhledávacího boxu tak, aby uživatel v reálném čase viděl návrhy frází, podobně jako je zobrazuje např. vyhledávání od Google. Princip našeptávačů obecně spočívá ve skriptu na straně klienta, který zadávaný text průběžně odesílá nějaké metodě na straně serveru, která vrací relevantní návrhy frází. Skript v prohlížeči pak tyto návrhy zobrazí formou v rozbalovacím seznamu.

Návrhy se generují v databázi, kde už byla připravena potřebná procedura. Potřeboval jsem ale vytvořit balíček na back-endu, pohled na front-endu a nakonec vymyslet skript běžící v prohlížeči, který bude pohled volat. Jelikož je debuggování skriptů v prohlížečích celkem neohrabané, vytvořil jsem nejdříve všechny ostatní části a otestovat je pomocí unit testů. Teprve když jsem si byl jistý, že fungují, začal jsem psát klientský skript. Jelikož celý web používá JavaScriptový framework jQuery, použil jsem jej i zde. Nejprve jsem zkoumal možnost použít funkci Autocomplete z frameworku jQuery. Připadala mi ale špatně přizpůsobitelná našim potřebám. Nakonec jsem tedy skript napsal jen s použitím obyčejné metody $.ajax.

Obrázek 3: Ukázka našeptávače

Návrhy frází jsem přenášel serializované do formátu JSON. Obvykle se kromě něj používá ještě XML, které se hodí především pro data se složitější strukturou, mezi kterými se pak dá navigovat pomocí jazyka XPath. JSON je zase úspornější objemem přenášených dat.

4.11 Přepínání měn

Platforma podporuje více měn, ve kterých může zákazník nakupovat. Tento úkol spočíval v implementaci boxu, pomocí kterého může zákazník měnu přepnout.

Na aplikační vrstvě stačilo připravit metodu, vracející seznam povolených měn. Zbytek implementace spočíval na front-endu.

Začal jsem samotným zobrazováním přepínače. Vytvořil jsem komponentu, která jej reprezentovala a XSL šablonu zajišťující její vykreslení. Do logiky komponenty jsem přidal funkci pro načtení povolených měn, která se volala při její inicializaci.

Poté jsem řešil složitější část – perzistenci zvolené měny. Jelikož ostatní komponenty a vůbec celá aplikace získávaly ID aktivní měny z objektu jménem WorkContext, musel jsem nové ID ukládat také tam. Jedno možné řešení spočívalo v přepsání vlastnosti WorkContextu přímo v akci, kterou uživatel vyvolá kliknutím na symbol měny. Takto by sice nové ID zůstalo zachováno po celou session, ale mohl by vzniknout problém při vykreslování první stránky, která přímo následuje po volbě měny. Jelikož nelze nijak nastavit pořadí zpracování komponent, mohlo by se stát, že některá komponenta přečte ID původní měny a teprve po ní dojde ke zpracování kódu přepínače, který ID aktualizuje.

Problém by bylo možné vyřešit pomocí jednoho dalšího přesměrování, díky kterému by už nová stránka spolehlivě pracovala s aktualizovanou měnou. To mi ale nepřišlo moc elegantní. Navíc, zpracování dvou požadavků by zřejmě trvalo déle, než zpracování jednoho, což by uživatelé určitě neocenili.

Vhodné řešení jsem našel až na úrovni HTTP modulu. Upravil jsem jej tak, aby uměl detekovat požadavek na změnu měny a uložil její ID do WorkContextu dříve, než se aplikace vůbec dostane ke zpracování kódu komponent.

4.12 Session Provider

ASP.NET poskytuje objekt Session, do kterého je možné ukládat objekty, které pak zůstávají perzistentní, jak už vyplývá z názvu, po dobu životnosti dané session. Můj úkol spočíval v návrhu vlastního řešení, které by umožnilo použít i jiná úložiště, než přímo paměťový prostor aplikace. Pokud by se data ukládala např. do společné databáze, bylo by možné větší projekty hostovat na více webových serverech současně.

4.13 Filtrování zboží v kategorii

Filtr zboží v kategorii je box, známy z většiny internetových obchodů, který umožňuje nějakým způsobem omezit nebo setřídit nabízené zboží. Mým úkolem bylo takový box naimplementovat.

Jelikož řešení vyžadovalo úpravy komplikovaných procedur v databázi, spolupracoval jsem s databázovým specialistou. Musel jsem vyřešit také způsob předávání dat mezi komponentou, reprezentující filtr a jinou komponentou, která zobrazovala vyfiltrované zboží.

Pro příjemnější práci se systémem z pohledu uživatele jsem filtrování řešil asynchronními požadavky, pomocí technologie AJAX. Změna kritérií nebo přepnutí nebo procházení výsledků hledání nevyžadovalo překreslení celé stránky. Řekl bych, že v míře využití technologie AJAX byl náš projekt celkově velmi na úrovni.

Filtr automaticky skrýval funkce, které nebyly v dané situaci relevantní. Pokud zboží v kategorii nemělo žádné atributy („novinka“, „akce“, „výprodej“. . .), pak se ani nezobrazoval jejich výběr. Podobně fungovala např. skladová dostupnost – bylo-li všechno zboží skladem, nezobrazil se filtr dostupnosti.

Pro efekt a lepší přehled uživatele, nad tím, co se na stránce děje, jsem průběh načítání nových dat animoval velkým točícím se kolečkem, které překrývalo zobrazené zboží a po dokončení požadavku zmizelo.

Ve finální verzi bylo možné zboží filtrovat dokonce podle různých ne zcela jednoznačně porovnatelných kritérií, jako jsou barvy, kdy jedno zboží může mít současně více barev, nebo velikosti, která se skládá z výšky, šířky a délky atd. Filtr si poradil také s veličinami v různých jednotkách – srovnatelné jsou třeba hmotnosti zadané v kilogramech i librách, délky v centimetrech, palcích, stopách i yardech a další.

4.14 Další úkoly

Kromě výše popsaných úkolů jsem průběžně řešil ještě i nějaké další, které ale byly zrušeny, přeplánovány do budoucnosti, podobné již popsaným problémům nebo jen nevýznamné:

- Modul dárky, který byl zajímavý možností definování velmi komplexních pravidel pro udělování dárků. Byl odložen.
- Opravy různých logických chyb na front-endové části, když se něco nechovalo podle specifikace.
- Psaní nových metod v modulech, když jsme dodatečně zjistili, že nějaké potřebujeme.
- Ladění CSS stylů, aby stránky vypadaly pokud možno stejně ve všech prohlížečích, se kterými náš projekt počítá.
- Refactoring ORM vrstvy kvůli změnám v databázi.
- Úpravy uložených procedur v databázi, které jsem čas od času dělal sám, když zrovna neměli databázoví specialisté čas.

5 Závěr

5.1 Uplatněné znalosti získané studiem

Řekl bych, že znalosti, které jsem získal během studia a uplatnil v praxi, obecně nebyly o konkrétních jazycích nebo technologiích. S těmi jsem víc zkušeností, než ve škole, získal během mých předešlých prací. Studium považuji za užitečné spíš v tom, že mi dalo různé teoretické základy, díky kterým jsem se na řešené problémy uměl dívat více v souvislostech. Za užitečné považuji především předměty *Vývoj informačních systémů*, *Databázové a informační systémy* a *Úvod do softwarového inženýrství*. Při tvorbě šablon na front-endu jsem si vzpomněl také na předmět *Uživatelská rozhraní*.

5.2 Chybějící znalosti

Novinkami, se kterými jsem se do té doby nesetkal, pro mě byla práce se sdíleným zdrojovým kódem a XSL transformace. Jinak si myslím, že mi žádné zásadní znalosti nechyběly. Se všemi ostatními technologiemi, které jsme používali (C#, ASP.NET, T-SQL, LINQ, HTML5, CSS, XPath, JavaScript a jQuery) jsem již nějaké, aspoň malé zkušenosti měl.

5.3 Zhodnocení praxe

Praxi v NetDirectu považuji za velmi přínosnou zkušenost. Jsem rád, že jsem pracoval právě ve vývojovém oddělení, jelikož jsem se tak podílel na vývoji moderního, velmi komplexního systému, jehož zkoumáním jsem si utřídil vědomosti ohledně návrhu informačních systémů.

Velkým přínosem mi určitě budou i zkušenosti s agilní metodikou Scrum a obecně s prací v týmu, která kromě spolupráce s kolegy obnášela také práci se sdílenými zdrojovými kódy.

Zvykl jsem si také správně používat unit testy, které jsem sice znal i předtím, ale používal jsem je neefektivně, jelikož jsem nebyl moc zběhlý v otázkách co, kdy a jak testovat.

6 Reference

[1] FOWLER, Martin, David RICE, Matthew FOEMMEL, Edward HIEATT, Robert MEE a Randy STAFFORD. *Patterns of Enterprise Application Architecture,* Addison Wesley, 2002. ISBN 0-321-12742-0.

[2] KOSEK, Jiří. *XSLT v příkladech.* [online]. [cit. 2012-03-14]. Dostupné z: `http://www.kosek.cz/xml/xslt/`

[3] FOWLER, Martin. Inversion of Control Containers and the Dependency Injection pattern. *Martin Fowler* [online]. [cit. 2012-03-24]. Dostupné z: `http://martinfowler.com/articles/injection.html`